Tablice

Piotr Zierhoffer

Institute of Computer Science
Poznań University of Technology

7 października 2012

Tablice

Tablice pozwalają na przechowywanie **wielu** obiektów, **tego samego** typu.

Pozwalają na przeprowadzenie operacji na zbiorach, kolekcjach itd.

Każda tablica posiada swój typ, adres i stały rozmiar. Każdy element tablicy to jej komórka.

Deklaracja tablicy

```
typ_elementu nazwaTablicy[rozmiar];
```

```
int buffer[10];
```

Tablica `buffer` zawiera w sobie 10 elementów typu `int`.

Typem zmiennej `buffer` jest **tablica int**.

Korzystanie z tablicy

Każda komórka tablicy posiada swój indeks.

Uwaga!

Tablice w języku C **zawsze** indeksujemy od zera!
Jeżeli tablica ma *n* elementów, to ostatni element ma indeks *n-1*.

Do elementów tablicy odwołujemy się podając indeks w nawiasach kwadratowych.

```
int tab[10];
int i = tab[4];
tab[7] = 5;
tab[0] = tab[1];
```

Przeglądanie tablicy

Najwygodniej przegląda się tablicę za pomocą pętli `for`.

Przeglądanie tablicy nazywamy *iterowaniem*, każdy krok pętli nazywamy *iteracją*.

```
int pierwsza[10], druga[10];
int i;
for(i = 0; i < 10; ++i)
{
  pierwsza[i] = i;
  druga[i] = i % 2;
}
```

Rozmiar tablicy

Czy poprawny jest poniższy zapis?

```
int tab[10];

int i = tab[-10];

for(i = 0; i <= 10; ++i)
{
  printf("%d\n", tab[i]);
}
```

Tak!
Taki zapis jest poprawny, ale czy ma sens? Czy jest bezpieczny?

Zadanie

Zadanie 1

Stwórz 10-elementową tablicę. Wypełnij ją liczbami od 1 do 10. Następnie wypisz jej zawartość na ekran.

Tablice a wskaźniki

Żelazna zasada na dziś

Nazwa tablicy jest adresem pierwszego jej elementu!

Te zapisy są równoważne!

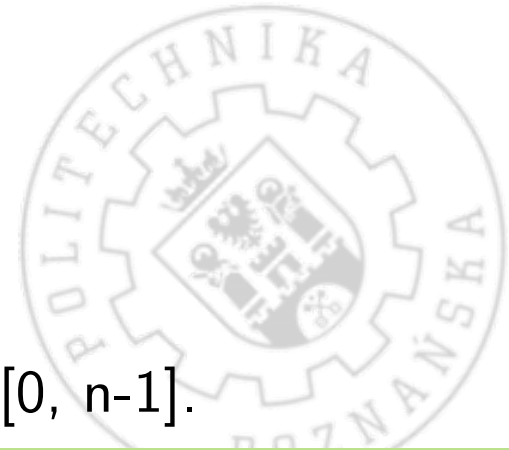
```
int tab[10];
int *ptab, i;

ptab = tab;
ptab = &tab[0];

i = tab[2];
i = *(tab + 2);

ptab[4] = 5;
*(tab + 4) = 5;
```

Do zapamiętania: nie przekraczać wymiarów tablicy: [0, n-1].



Przekazywanie tablicy jako parametr funkcji

Jeżeli `tab == &tab[0]`, to znaczy, że `tab` jest zawsze interpretowane jako wskaźnik!

```
void fun(float *tab, int size)
{
  ...
}

int main()
{
  float tablica[10];
  fun(tablica, 10);
}
```

Funkcja zazwyczaj nie zna rozmiaru tablicy, należy go przekazać jako parametr.

Zadanie

Zadanie 2

Zmodyfikuj zadanie 1 tak, by wszystkie operacje Z WYJĄTKIEM deklaracji tablicy działy się w funkcji `tab_fun`.

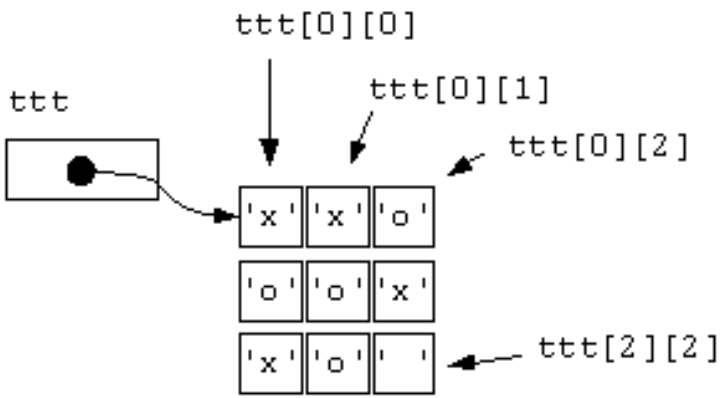
Tablicę zadeklaruj w funkcji `main`.

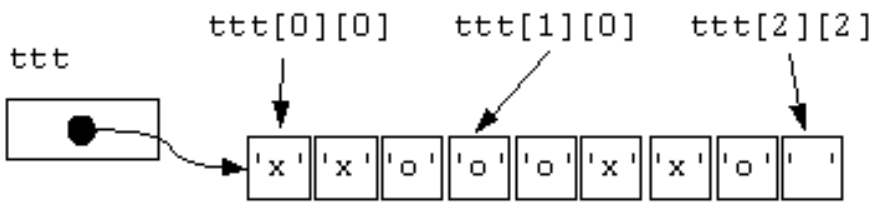
Tablice wielowymiarowe

Tablica jednowymiarowa to “ciąg” zmiennych.

Tablice wielowymiarowe to prostokąty, sześciany, ...

```
int tab[3][4];
float tab2[40000][2][50];
```





Rys.: Tablica dwuwymiarowa

Napisy

Napis (ang. *string*) to tablica znaków zakończona znakiem '\0'.

```
char *tab = "Jakis napis";
'J','a', 'k', 'i', 's',' ', 'n', 'a', 'p', 'i', 's','\0'
```

Język wspiera wykorzystanie napisów. Wiele użytecznych funkcji dostępnych jest przez nagłówek `string.h`: badanie długości, wyszukiwanie, **kopiowanie**, porównywanie, łączenie.

Znak '\0' mówi o zakończeniu łańcucha - pozwala m.in. na wyznaczenie jego długości.

Wypisywanie i wczytywanie:

```
char * tab = "Moj napis";
char tab2[10];

printf("%s\n", tab);
scanf("%s", tab2);
```

Zadanie

Zadanie 3

Wczytaj napis do tablicy znaków (co najmniej 20 znaków).

Wypisz na ekran jego długość.

Zamień pierwszych 5 znaków na ciąg “AAAA”.

Wypisz powstały napis na ekran.

Inicjalizacja

1. Tablica jednowymiarowa

```
float tab[10] = {0.0, 1.1, 2.2, 3.3, 4.4, 5.5,
      6.6, 7.7, 8.8, 9.9, 10.10};
float tab[] = {0.0, 1.1, 2.2, 3.3, 4.4, 5.5,
      6.6, 7.7, 8.8, 9.9, 10.10};

int tab[10] = {0};
```

2. Tablica wielowymiarowa

```
int tab[3][2] = { 0, 0, 1, 1, 2, 2};
int tab[3][2] = { {0, 0}, {1, 1}, {2, 2}};
int tab[][2] = { {0, 0}, {1, 1}, {2, 2}};
```

3. Napisy

```
char *p = "Napis";
char tab[] = "Napis";
```